EUROGRAPHICS '93 R. J. Hubbold and R. Juan (Guest Editors), Blackwell Publishers
© Eurographics Association, 1993
Volume 12, (1993), number 3

A Dynamic Gesture Language and Graphical Feedback for Interaction in a 3D User Interface

Monica Bordegoni (1)(2) and Matthias Hemmje (1)

(1) IPSI-GMD, Dolivostrasse 15, D-6100 Darmstadt - Germany
(2) IMU-CNR, Via Ampere 56,20131 Milan - Italy
e-mail: [bordegon, hemmje]@darmstadt.gmd.de

Abstract

In user interfaces of modern systems, users get the impression of directly interacting with application objects. In 3D based user interfaces, novel input devices, like hand and force input devices, are being introduced. They aim at providing natural ways of interaction. The use of a hand input device allows the recognition of static poses and dynamic gestures performed by a user's hand.
This paper describes the use of a hand input device for interacting with a 3D graphical application. A dynamic gesture language, which allows users to teach some hand gestures, is presented. Furthermore, a user interface integrating the recognition of these gestures and providing feedback for them, is introduced. Particular attention has been spent on implementing a tool for easy specification of dynamic gestures, and on strategies for providing graphical feedback to users' interactions. To demonstrate that the introduced 3D user interface features, and the way the system presents graphical feedback, are not restricted to a hand input device, aforce input device has also been integrated into the user interface.

Keywords: Interactive techniques, novel graphic applications, novel input devices.

1. Introduction

Some user interfaces of today's computer applications require the presentation of data in various media such as text, video, complex graphics, audio, and others. The effort for giving a realistic appearance to information data aims at simplifying users' tasks, yielding them more natural and close to users' habits and skills. On the one hand, information from the system should be immediately captured by users without any cognitive costs for interpreting and understanding it. On the other hand, information should be easily transferred from users to the system.
Whenever possible, information may be presented in the same way people would perceive it in the real world. In case of abstract data, a representation should be good enough to communicate as much information as possible. To achieve this goal, spatial metaphors for data presentation seem to work quite successfully.
User interfaces of modem systems are becoming more and more transparent. This means that users get the impression of directly interacting with application objects, rather then doing it via a computer. Especially in 3D based user interfaces, traditional 2D input devices are no longer adequate for supporting these kinds of

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

interaction, as, e.g., they do not support concepts like spatial depth. Therefore, more powerful and expressive devices are required. Current technology is proposing novel input devices, such as flying mouse, spaceball, glove, etc., to fulfill this task, Some of them try to provide natural ways of interaction, which are more close to human habits of expressing thoughts and interacting with their surrounding world.
This paper describes the integration of a hand input device based on the requirements of applications using 3D user interfaces. We have developed a dynamic gesture language, a graphical tool for its specification and a gesture recognition system.This system recognizes dynamic gestures, when performed by a user wearing a hand input device, and sends information about recognized gestures to a 3D application. Moreover, it provides a helpful and meaningful graphical feedback to user's input. To demonstrate that the introduced 3D user interface features, and the way the system presents graphical feedback are not restricted to a hand input device, a force input device has also been integrated into the user interface.

2. Motivations

Nowadays, many user interfaces which make use of spatial metaphors[1]are developed. The goal of our work is to define a suitable way of interacting with such user interfaces based on three-dimensional visualizations of the application domain. At first, we outline general requirements and properties of such interactions.
While interacting with a 3D user interface, the users' dialogue with the system consists of mainly navigational interaction like e.g. changing view and position, zooming in/out. etc. These are taking place within the user interface's virtual 3D space. Furthermore, there are actions like selecting, grabbing, moving and turning graphical objects, retrieving information by querying objects, introducing some commands (undo, browsing commands, etc.). For all these types of interactions users have to be provided with a feedback, to confirm that the system has received their input. By examining potential applications like for example [1], 3D CAD Systems, etc., we identified the following set of basic interactions:
- navigation: change view and position in space;
- picking: select an object;
- grouping: group objects;
- querying: visit objects content;
- zooming in/out: change distance between objects and user's point of view;
- grabbing, rotating, moving: change objects position in space.

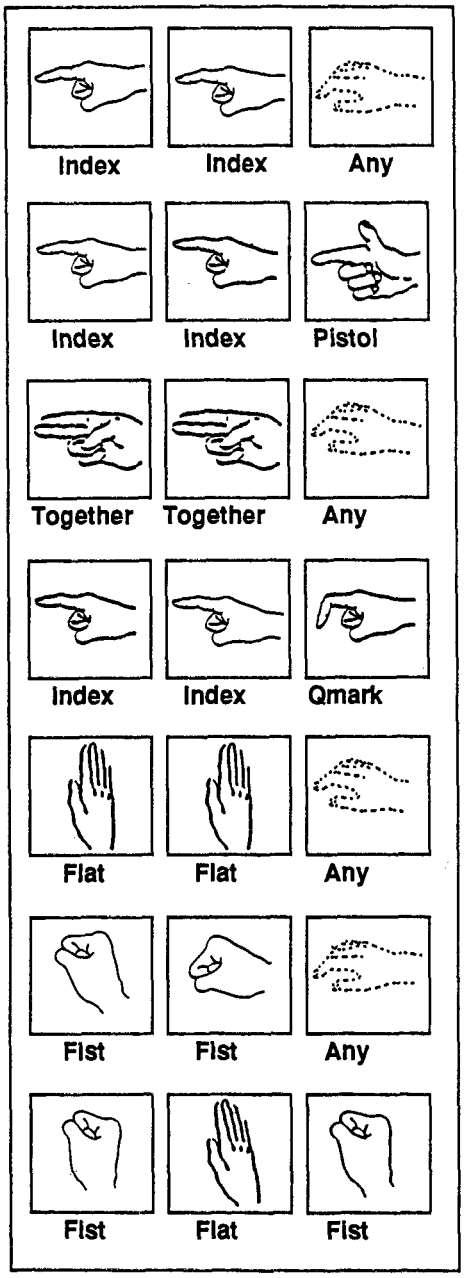
Given the 3D nature of the application, traditional 2D input devices, such as mice and tablets, seem no longer adequate to implement these interaction functionalities. More powerful and expressive devices, that easily support 3D interaction, are required [2]. To provide user interfaces with above outlined functionality, we have decided to choose the two input devices that are most appropriate[3][4]: a hand input and a force input device. In the following, we introduce a user interface which takes advantage of the capabilities of these input devices and, at the same time, implements the above characterized way of interaction.

3. Gesture Based Interaction

We define a pose as a static posture of the hand Characterized by bending values of joints and orientation of the hand. Our approach extends this capability providing the recognition of *dynamic gestures*. Dynamic gestures are powerful in that they allow humans to combine a number of poses and easily communicate complex input messages quasi in parallel. For example, it is possible to specify an object, the operation to perform on the object and additional parameters by means of one dynamic gesture.
We introduce a *dynamic gesture language* as a means of interaction, as well as a method for dynamic gestures recognition.

3.1 The Dynamic Gesture Language

The gestures chosen for interaction with the application have different features, so that on the one hand users can perform them easily, and on the other hand the system is able to recognize them undoubtedly. This is achieved by using *poses* and their *trajectories.* We determine a dynamic gesture by a *sequence of poses* performed over a particular trajectory. In the following, gestures of the language suitable for a 3D application are described. The defining sequences of poses are listed accordingly in Figure 1.



Navigation gesture
The application starts performing a navigation task when the "Index" pose is performed. A rotation of the hand changes the point of view of the 3D scene. When the pose is released, the gesture is over.

Picking gesture
During navigation, when an object, or a set of objects, are reached, they can be selected by performing the pose "Pistol".

Grouping gesture
The gesture starts with the "Together" pose. The user then needs to draw with the hand the diagonal of a bounding box limiting the objects to group. The gesture finishes when the pose is released.

Querying gesture
This gesture starts with the "Index" pose, too. When an object is reached, its content can be visited by performing the "Qmark" pose, which is the final pose of the querying gesture.

Zooming gesture
This gesture starts with the "Flat" pose performed with the back of the hand towards the user. If the hand is moved away from the user, a zooming in task is performed; if it is moved towards the user, a zooming out task is performed.

Gripping gesture
This gesture starts when the "Fist" pose is performed. The object is grabbed, rotated and moved until the "Fist" pose is released.

Exit gesture
The gesture simulates a good-bye wave. This consists of opening and closing the hand, with the back of the hand towards the user ("Fist" pose, followed by a "Flat" and then by a "Fist" pose).

Figure 1. *Poses Compounding Gestures of the Language*

3.2 Gesture Specification

On the one hand, teaching and recognizing very complex gestures is a non trivial task [5][6][7], on the other hand, the considered applications do not require very complex gestures. We decided to concentrate on an approach that enables the user, or the system designer, to easily teach the system a new gesture, by using sequences of poses.
Having studied the composition of gestures appearing in our language, we have identified the poses featuring in the whole gesture set. During our experiments, we revealed that six basic poses are sufficient to define the above described gestures. Every user of the system can teach this set of poses easily to the hand input system, using the Dynamic Gesture Editor.
A *Dynamic Gesture Editor* provides users with some facility for the definition of gestures by combining the selected poses and setting their characteristic values (orientation, trajectory, etc.).
For defining a new gesture, users have firstly to identify the main features of the gesture. Then, they have to describe these features, by selecting a sequence of postures from the menu. If further postures are necessary, they can be added to the menu by teaching them to the system. Finally, every posture of the gesture has to be associated with an orientation and trajectory value. It is also possible to associate a cursor with each defined gesture. It will be used by the system for providing feedback to the performed gesture, as described in section 4. Figure 2 shows, as an example, the definition of the Exit gesture. After defining the three postures composing the gesture, an orientation value of the hand can be defined for each posture.

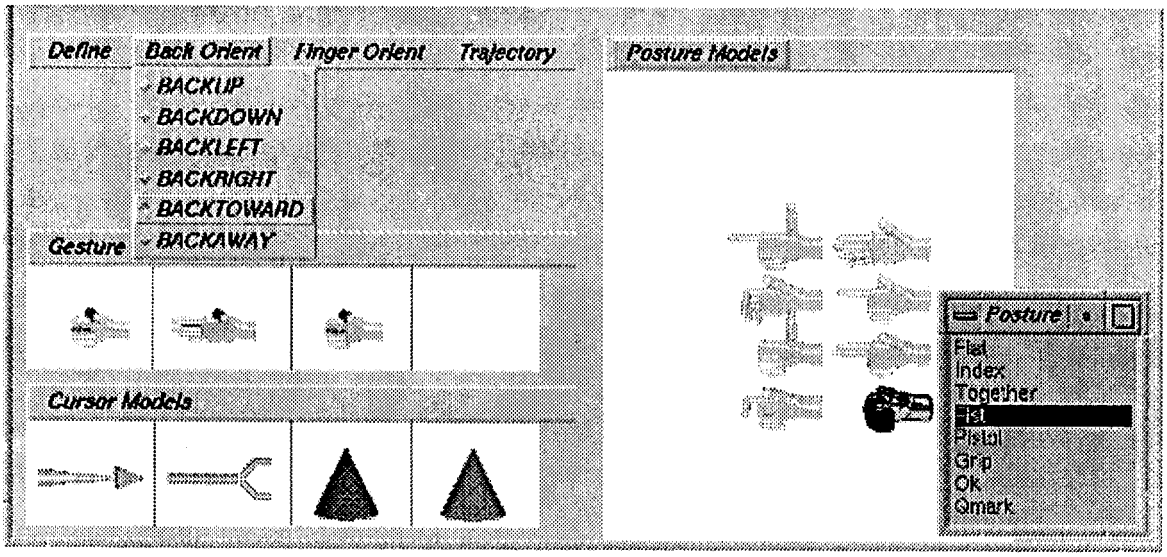


Figure 2. *Dynamic Gesture Editor*

To see and test the new defined gestures, the editor provides a *simulation functionality* which dynamically reproduces the defined gestures. Newly taught gestures are stored in a database of *Gesture Models,*
The main advantage of this approach is that users do not need to physically perform gestures for teaching them, Another advantage of this approach compared to e.g. Neural Network approaches [5][6] is that less efforts have to be spent on training (wether manpower or computational). Users only need to combine predefined poses with orientation and direction values. It is like composing words, given some letters of an alphabet. Another advantage is that a gesture language can be defined by a single user and then used by many users.

3.3 Characteristics of Gesture Recognition

In the following, we highlight gesture characteristics important for their recognition. These characteristics specify the relevance of static postures, orientation and trajectory for the recognition of each of the gestures. Moreover, the characteristics determine the importance of detecting all poses forming a gesture as well as the accuracy with which a gesture is recognized and also its length in time.
Finally, table 1 summarizes the setting of the characteristics of the gestures described above.

Hand posture. The posture of the hand may change during the performance of the gesture. For example, the gesture Picking consists of the initial pose "Index", the final pose "Pistol" and all poses in between.
In other cases, the hand posture is always the same over the all gesture. Some pose sets the end of the gesture, like the Navigation and the Zooming gestures. Using a general pose for ending a gesture is also useful in situations where the user needs to be able to disengage from the task or suspend input.
Poses orientation detection. Each pose of the gesture has an orientation. For the recognition of the gesture, this orientation can be negligible or not. This has to be determined in the definition of the gesture.
For example, in the Navigation gesture, the orientation of the hand is important, as it affects the user's point of view within the scene. In the Gripping gesture, setting in advance the Orientation that the hand has to hold during the gripping, causes an unnatural constraint to the user.
If the gesture is used for navigating in a room, where the user can only "walk" on a "floor", the system provide some ways to eliminate unwanted degrees of freedom. So the user is no longer trying to avoid motion in these degrees of freedom.
Trajectory detection. In some gestures, the detection of the trajectory is not useful or desired, while it may be important in others. This has to be determined, too. If the user wants, e.g., to grip a 3-D object and move it within space, the trajectory detection is not important. The system has to be detect the action for catching the object and assume the hand's position and orientation as parameters of the gesture. These are used for positioning the object in space, but not for defining the gesture. In the Zooming gesture, the detection of the trajectory is important for deciding if the intent is zooming in or out the scene.
Middle poses detection. Middle poses are all poses occurring between the first and the last pose of a gesture. Sometimes, checking the correctness of all middle poses of a gesture, may be of no interest. In other cases, the entire sequence of poses is relevant for the characterization of gestures, and therefore it needs to be checked. An example of the first case is the Gripping gesture. The system has to know the initial pose (picking up the object) and the final pose (releasing the object), but does not need to know anything about the sequence of poses in between.
Confidence factor. During the recognition of gestures, it happens that for some reasons (related to human capability of reproducing gestures with accuracy or to recognition algorithm inaccuracy), a part of the performed gesture does not match the model. The confidence factor of a dynamic gesture defines the percentage of recognized poses, over the total number of poses that needs to match, so that the gesture is recognized. As gestures used by our system are simple and poses have no similar features, gestures are expected to be recognized with high accuracy (the percentage is expected to be close to 100%).
Gesture duration. Sometimes, it is impossible to predict in advance the duration of a gesture. For example, in the Navigation gesture, the gesture lasts as long as the user reaches an object or a proper view of the scene. Some other gestures, like Grouping and Exit, may require a duration of only a few seconds.

Table 1: *Setting of characteristics for the introduced gestures*

	Hand Configuration	*Orientation Detection*	*Trajectory Detection*	*Middle poses Defection*	*Gesture Duration*
Navigation	Index -> Any	no	no	no	off
Picking	Index -> Pistol	no	no	no	off
Grouping	Together -> Any	yes	yes	no	3 secs
Querying	Index -> Qmark	no	no	no	off
Zooming	Flat -> Any	yes	yes	no	Off
Gripping	Grip -> Any	no	no	no	Off
Exit	Flat -> Fist	yes	no	yes	1 sec

3.4 Gesture Recognition

The system includes a module named *Gesture Machine* [7][8], which checks if data are satisfying the model of one of the gestures stored in the database.
As outlined, each gesture model is defined as a sequence of poses, where each pose is described by hand's finger flexion values, orientation and trajectory value. The algorithm used by the Gesture Machine works as follows.
When a new input pose arrives, the Gesture Machine checks if it matches the starting pose of one or several gesture models. If a match occurs, the corresponding gestures are set to be *active.* An *Actor* object is associated with each active gesture. It keeps the history of the gesture and updates a pointer to the currently expected pose.
When a new pose arrives, it is required to match the expected pose or the previous one. When all poses of a model, or a percentage of them according to the Confidence Factor defined for the gesture, have been recognized, the gesture as a whole is set to be recognized.
A parameter sets the number of consecutive mismatched poses over which the gesture is not recognized any more. If the expected pose is B and the previous is A, some poses are detected by the system while the hand performs the movement from pose A to B. The system discards a number of *noisy poses* up to the number of allowed consecutive mismatches.
The application is constantly informed about the position and orientation of the hand and of the gestures recognized. This information is useful to perform transformations on application objects and to provide output according to user's interaction.
Some examples of poses and gesture recognition are shown in Figure 3.

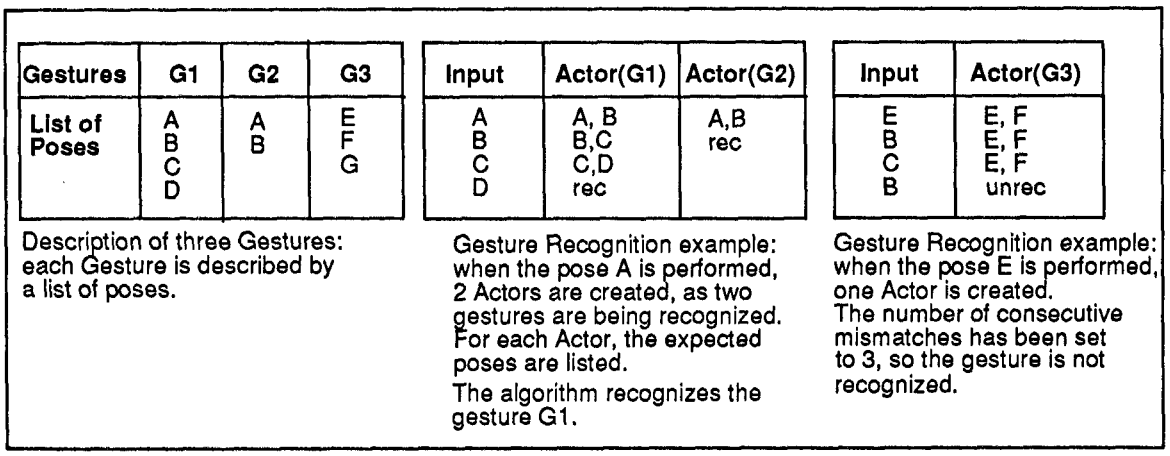


Figure 3. *Example of Gesture Recognition*

4. Gesture Feedback

During our experiments, we recognized that while interacting in a 3D based user interface, it is very important for the users to get a helpful feedback. Otherwise users can not estimate wether their input has been realized by the user interface, Changes performed over the device needs to be constantly monitored. Moreover, a semantic feedback to the actions performed by users is also very important, to make sure that the system did not only receive the input but also is interpreting it correctly. Therefore, our system provides three types of feedback: a graphical hand, some virtual tools and graphical changes over the objects of the scene. Furthermore, this chapter outlines how non-hand input devices can also benefit from the gestures and the feedback concepts described in the following.

4.1 Graphical Hand

In our user interface, a graphical hand provides a natural feedback to the user's "real" hand. The graphical hand moves according to user's hand movements within the application space and the graphical hand reflects every movement of finger's joints. When a gesture is being recognized, the color of the hand changes. Different colors can be associated with different gestures. In the following sections, we will outline how the intuitiveness of the feedback has been further improved.

4.2 Virtual Tools

During the performance of particular actions, like e.g. the picking gesture, the hand as a cursor has not always appeared to be precise and accurate enough for achieving the task. In such cases another kind of graphical

feedback is more appropriate. A first attempt for identifying a suitable feedback, has been done with the Navigation gesture. If users want to reach an object for querying its content, they should be able to reach it easily and with precision. If graphical objects are small, the graphical hand can partially or totally obscure their view. A feasible approach is to adopt the metaphor of *hand as a tool* [9]. The hand can assume the feature of a virtual tool, more suitable for the specific task.
This approach serves the purpose of giving a semantic feedback to user's action by showing a tool commonly used (in real life or in the computer field) for achieving that task. Moreover, it is possible to avoid showing hand's degrees of freedom that are not proper of the tool and not required in the task.
In our prototype, when the Navigation gesture is being recognized, the cursor appears as a small arrow: the object is reached when the head of the arrow "touches" it. Another cursor has been defined for the Gripping gesture. In this case, some pincers are used in place of the graphical hand. When a gesture stops being recognized, the hand feedback returns to its normal hand shape.
Pictures at the end of the paper visualize some examples of feedback provided by the system*. The two pictures on the left show the rendered hand displayed when no gesture is recognized. The upper-right picture depicts some pincers displayed when the Gripping gesture is performed. The lower-right one shows an arrow pointer visualized when the Navigation gesture is performed.

4.3 Object Reaction

In some cases, feedback can be performed over the object affected by the action, instead of changing cursor shape or color. For example, the picking gesture is fast, so that a feedback performed over the cursor would be hardly noticed. It is better to visualize the success of the action by changing the color of the picked object. In opposite, the query gesture requires a feedback, as the response from the database could take few seconds.
As the structure of graphical objects of the scene is known only by the application, and not by the hand input system, it is up to the application to provide feedback on its graphical objects, for reacting to user's input.

4.4 Porting the Concepts

To demonstrate that the introduced 3D user interface features, and the way the system presents graphical feedback are not restricted to a hand input device, a force input device has been integrated into the same 3D application.
Force input devices are more precise than hand input devices for reaching a specific location in space. They perform well for pointing at objects when these are small and many in the scene.
To use the application with a force input device as well, an attempt to map the gesture language into a language for this device has been done successfully. Buttons of the force input device can be used to perform actions.
The main problem for the users when using force input device buttons for interacting, is that it is easy to forget which is the button that needs to be pressed to perform an action. Associating an action with each button is successful only if the user interface provides some help showing the proper correspondence. In our application, a button of the device switches between Navigation and Zooming action: while navigating, the cursor moves in the scene; while zooming, it is the scene that is moved and scaled. An object is picked, queried or otherwise manipulated by selecting appropriate buttons. To support the user in the choice of buttons, the cursor reacts in the same way as described in the previous chapter, by graphical feedback to e.g., changing between different tool shapes.

* **See page C–517 for Colour Plate.**

5. User Interface

This section describes the user interface architecture, shown in Figure 4, integrating the interaction devices, the graphical interface and our modules for gesture recognition and graphical feedback.

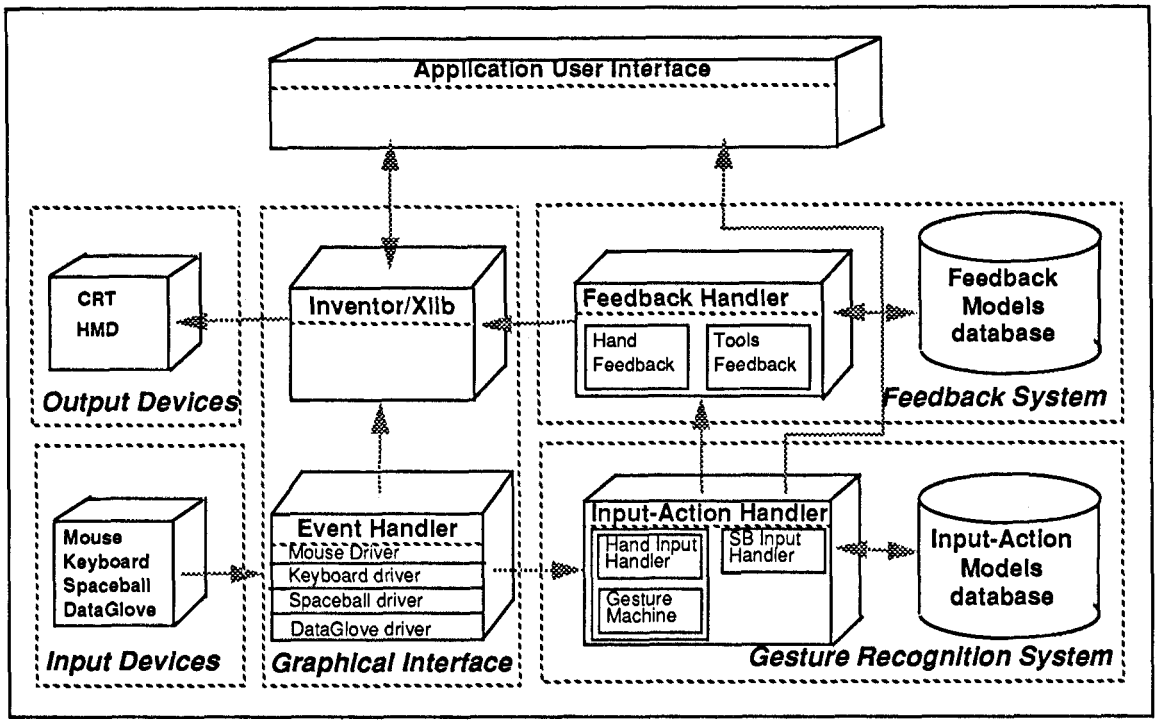


Figure 4. *User Interface Architecture*

5.1 Interaction Devices and Graphical Interface

Graphical Interface. The graphical interface is provided by Silicon Graphics Iris-Inventor, an object oriented 3D toolkit based on [10] and running on top of an X Window System. It allows rapid prototyping of 3D visualizations with low implementation effort on the one hand, and takes advantage of powerful graphics hardware features on the other hand. The application user interface as well as the Feedback and the Gesture Recognition Systems described below, communicate their visualization requests to this module.

Interaction devices. Among the available devices, we have chosen to use the Spaceball [11] and the VPL DataGlove [12]. The *Spaceball* measures the intensity of the force exerted on the ball for providing 3D movements. It is supplied with a button on the ball itself and eight other buttons located on the device in a place easily reachable by user's fingers. The *VPL DataGlove* is supplied with a Polhemus device [13] for detecting orientation and position of the hand. Two sensors per hand's finger detect the bending of the first and second joint of each finger. Using some functionality of the VPL DataGlove system, it is possible to calibrate the glove for the specific user's hand and teach the system up to 10 poses that it may recognize [14]. The Spaceball as

well as mouse and keyboard are already supported by the X Window System and therefore are also integrated within the graphical interface. In addition, we have developed an appropriate integration of the Data Glove. The graphical output is visualized on either a high resolution CRT or a head-mounted display.

5.2 Gesture Recognition and Feedback Systems

Gesture Recognition System. The Gesture Recognition System consists of the *Input-Action Handler* and a database for Input-Action Models. The Hand-Input Handler on the one hand supplies the Gesture Machine with the necessary data for gesture recognition, and on the other hand transmits them to the application user interface. Data received from the Spaceball is checked by the SB Input Handler and also transmitted to the application user interface. In this way, both handlers recognize user's actions that match the Action Models stored in the *Input-Action Models database* and communicate corresponding requests to the Feedback Handler, to visualize the appropriate feedback model.
The system provides an interface which translates gesture identifiers used by the system into high level event codes used by the application. In this way, the application is independent from the gesture language. Each user can define his/her own language for interacting with an application. Moreover, an already defined language, or some "words" of it, can be used for interacting with other applications.

Feedback System. According to the requests the feedback system receives from the Input-Action Handler, appropriate feedback models from the Feedback Models database are retrieved and visualized by the Feedback System. To achieve this, the Feedback Handler requests either the Hand Feedback module or the Virtual Tools feedback module to perform this action.

6. Conclusions

This paper has presented the study of interaction in a 3D based user interface, performed by user's dynamic gestures and the interface's graphical feedback. In current state of the system, users can teach the system some gestures by means of a gesture editor. When these gestures are then performed by a user wearing a hand input device, a gesture recognition system recognizes them. It is also possible to interact in the same way by using a force input device. The system provides a feedback to users' interaction by means of changing cursor shape or color. This way of providing semantic feedback has revealed to be helpful for users' interaction with three-dimensional visualization of the application domain.
The study will proceed evaluating the performance of this way of interaction when used in very complex scenes. Moreover, we shall analyze if more complex hand gestures can be reliably detected by the recognition algorithms and wether they improve the intuitiveness of the interaction.

7. References

1. Card S.K., Robertson G.G., Mackinlay J.D., *The information Visualizer, an Information Workspace,* in Proceedings CHI'91, New Orleans, April 1991, ACM Press, p. 181.
2. McAvinney P., *Telltale Gestures - 3-D applications need 3-D input,* BYTE - July 1990, pp.237-240.
3. Felger W., *How interactive visualization can benefit from multidimensional input devices,* Alexander, J.R. (Ed.): Visual Data Interpretation, Proc. SPIE 1668, (1992).

4. Jacob, R.J.K., Sibert, L.E., *The Perceptual Structure of Multidimensional Input Device selection,* in Proceedings CHI '92, pp. 211-218.
5. Murakami K., Taguchi H., *Gesture recognition using Recurrent Neural Networks,* ACM 1991, pp. 237-242.
6. Fels S.S., *Building Adaptive Interfaces with Neural Networks: the Glove-Talk Pilot Study, University of Toronto,* Technical Report CRG-TR-90-1, February 1990.
7. Bordegoni M., *Dynamic Gesture Machine,* RAL, Report 92-019, Rutherford Appleton Laboratory, Chilton, England, February 1992.
8. Bordegoni M., *Dynamic Gesture Machine: un sistema per il riconoscimento di gesti,* Proceedings Congresso Annuale AICA, October 1992.
9. Prime M.J., *Human Factors Assessment of Input Devices in EWS,* RAL, Report 91-033, Rutherford Appleton Laboratory, Chilton, England, 1991.
10. Strauss P.S., Carey R., *An Object-Oriented 3D Graphics Toolkit,* Computer Graphics, 26, 2, July 1992, pp. 341-349.
11. Spaceball Technologies Inc. 1991.
12. Zimmerman T.G., Lanier J., Blanchard C., Bryson S. and Harvill Y., *A Hand Gesture Interface Device,* CHI+GI, 1987, pp. 189-192.
13. 3 Space user's manual, Polhemus - A Kaiser Aerospace & Electronics Company, May 22, 1987.
14. VPL Research Inc., *DataGlove Model 2- Operation Manual,* CA - USA, August 25, 1989.